Programming AVR Microprocessors

easy with BASCOM AVR

By Mark Alberts

This article shows that working with microprocessors has become very straightforward these days. Thanks to the AVR-microprocessor and the well-known BASIC programming language success is assured.

There are several different AVR-microprocessors available. The AT90S2313 is an inexpensive 20-pin version with 2 kB of flashable code memory. Flash memory is the area where the program is stored. The main advantage of this type of memory is that it can be reprogrammed a thousand times. An even bigger advantage is that you do not have to bother with special programmers or UV erasers — a simple cable connected to the LPT (parallel printer) port of your PC is all you need to program a flashable micro.

The 90S2313

First, let's have a look at the most important characteristics of the 90S2313 (**Figure 1**):

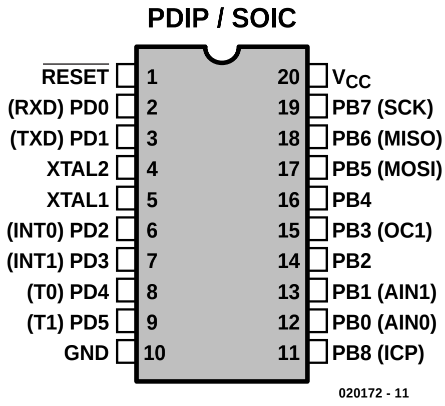


Figure 1. The connection details of the 90S2313.

There are two available ports: PORTB and PORTD
Internal memory: 128 bytes
EEPROM memory: 128 bytes
Flash program memory: 2 kBytes

A noticeable feature is that 15 of the 20 pins have been used for IO ports. It is these pins that provide the measurement and control capabilities of the processor. The ports normally consist of 8 pins (bits) each. However, pin PD7 on the 90S2313 is not actually connected to a physical external pin.

Any pin can function as an input or an output. If the pin is functioning as an output, it can be assigned a logic level of ‘0’ or ‘1’. The processor can do this for each pin individually, but it is also possible to address them as a group. Pins PD0 to PD6 together form the group called PORTD. By assigning a value to PORTD we can change the value of all pins in one go.

Most of the pins also have an alternate function. The alternate function is shown in brackets. *(INT0) PD2* means: the pin name is PORTD.2 and the alternate function is INT0. For the purpose of this article we will limit ourselves to the normal functionality of the pins.

Working with BASCOM AVR

Before we can actually get started, we first have to install the programming software. The software is called BASCOM-AVR (**BAS**IC **COM**PILER) and can be found on the *Elektor Electronics* 2002 Freeware-CD-ROM. Alternatively, download the files from the MCS Electronics web site at

www.mcselec.com/elektor.htm

where you can read which files you need to download, as well as find

Listing 1

```
CONFIG PORTB = OUTPUT ‘ use port B pins as output pins
PORTB.0 = 1 ‘ pull pin PORTB.0 high
PORTB.0 = 0 ‘ pull pin PORTB.0 low again
PORTB = 255 ‘ pull all pins high
PORTB = &B10101010 ‘ make pins 1,3,5 and 7 high
END
```

instructions on how to install the software. The software that has been used here is a demo version, which is however fully functional. The only limitation is that the compiled processor code is limited to 2 kB in size — the same as the flashable memory of the 90S2313!

Start BASCOM AVR and select *Files* ? *New* from the menu. A window will appear. In this window we will type our first program (**Listing 1**).

Save the program with *File* ➛ *Save As* and give it the name **sample1.bas**.

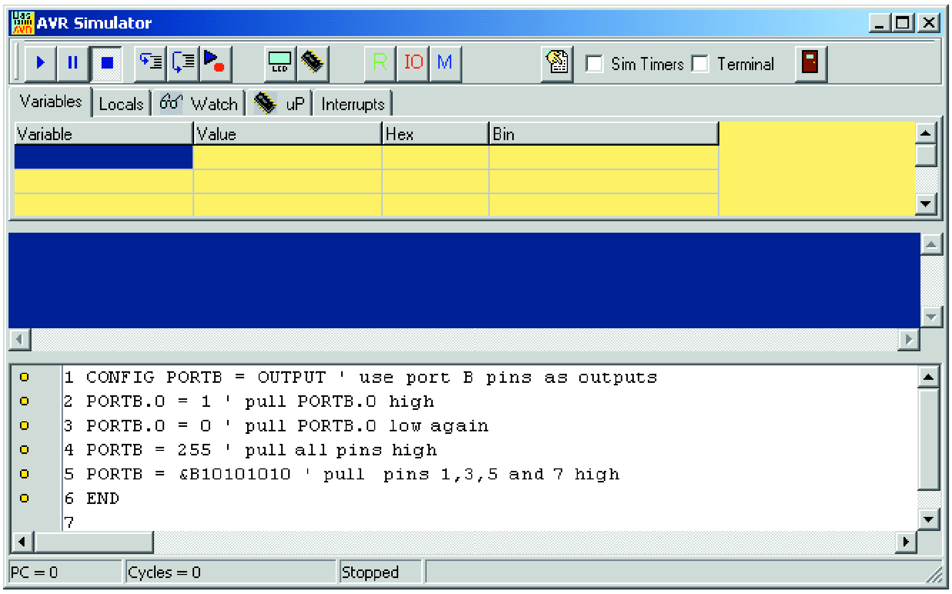
You've now written your first program. The microprocessor will understand very little of this program however, it can only understand the so-called object code. Object code is generated by compiling the BASIC program. Press F7 to accomplish this.

If you do not get any error messages, then you have entered the program correctly. If you do get an error message however, correct the mistake and compile the program again.

Before you program the microprocessor with the generated object code, you will first have to simulate the program. Only if you are satisfied with the results of the simulation, the program may be loaded init the microprocessor.

When no more error messages pop up, you may press F2 to open the simulation window.

The program as entered is shown at the bottom of the window (**Figure 2**). At the top of the window are the buttons controlling the simulation. Click the 'LCD' button in order to open the hardware simulation window.



```
1 CONFIG PORTB = OUTPUT ' use port B pins as outputs
2 PORTB.0 = 1 ' pull PORTB.0 high
3 PORTB.0 = 0 ' pull PORTB.0 low again
4 PORTB = 255 ' pull all pins high
5 PORTB = &B10101010 ' pull  pins 1,3,5 and 7 high
6 END
7
```

Figure 2. The simulation window of Bascom AVR.

There is now an LCD-panel at the top of the hardware-simulator window (**Figure 3**). Ports PORTB and PORTD are shown at bottom left. The purpose of the green LEDs will be explained later. We can now go through our program step by step. By clicking the step button (<), some invisible code is executed. The purpose of this code is to prepare the microprocessor for use. Once you click the step button again, the first line of the program is executed. This line 1 has no visible effect. Click the step-button again to execute line 2. LED 0 of PORTB will now light up.

This is because with 'PORTB.0 = 1' we gave the instruction to make PIN 0 of PORTB a '1'. This simulator indicates a logical one at a pin as an illuminated LED. The outcome of the following step will not surprise you: the LED goes out.

Line 4 causes all pins to go '1' by writing to PORTB. With this instruction we can change the level of all the pins in one go.

Line 5 is interesting because it shows that we can also use binary notation to provide the value of the port. The binary notation is prefixed with &B, followed by all the bit values. The first bit corresponds to pin 7, the last one to pin 0. For example, 'PORTB=&B1001' makes only pins 0 and 3 high.

Now that the program works, we can program it into the microprocessor. Press F4 to make this happen. After programming is completed, you will see that LEDs 1, 3, 5 and 7 are lit. Because the program is executed so rapidly by the microprocessor, you will not see the result of lines 2 through 4!

Naturally, it is possible to replace the LEDs with a relay or some other output circuit.

You may now like to know what the purpose of the green LEDs in the simulation window is. The AVR-microprocessor is also able to determine the logic level of a pin when the port is configured as an input. In this case the ports are called PINB and PIND.

Listing 2

```
CONFIG PORTB = INPUT
PORTB = 255
DO
   PRINT PINB
LOOP
```

Next, you remove the END statement from the program and add the following lines: Compile the program (F7) and press (F2) again to start the simulator.

Then press the Run button (>). You will see that in the centre of the screen the value of PINB is displayed. This is '0'. By clicking the green LEDs of PINB you can change the levels of PINB in the simulator. You will see that the value changes accordingly.

There is also a switch in our circuit. It is connected to PINB.0.

You now enter a new program:

Listing 3

```
CONFIG PINB.0 = INPUT  : CONFIG
  PINB.2 = OUTPUT : PORTB.0 = 1
DO
     PORTB.2 = PINB.0
LOOP
```

Compile this program and program the micro using F4.

If you now press button S1, the LED will turn off. Pin 2 of PORTB is now made equal to the logic level at input pin 0 of PINB.

In the BASCOM help file you can find the effect and explanation for each of the statements that were used here.

(020172-1)

Note:
The author of this article, Mark Alberts, is the designer of the BASCOM compilers for the 8051 and AVR-processors. More information can be found by visiting his website at www.mcselec.com.

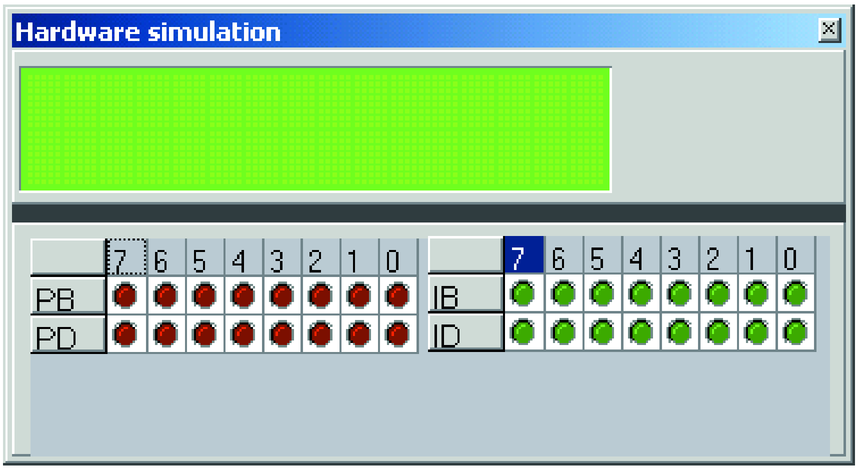


Figure 3. The hardware simulator consists of an LCD section with below it the input and output levels of the two ports.

Test example

The schematic shown here may be used to check various things.
Using the LPT connector, the so-called Sample Electronics programmer may be made. The series resistors are included for protection. It should be noted that it is best to connect the hardware via a separate IO card.

The DB9 connector, together with the MAX232, forms a serial input and output that can be connected to the COM port of a PC.

In the example, a switch is connected to PB0. SV1 is a header on which jumpers may be placed. The driver chip (ULN) is necessary in this case. Normally this IC is not required when the LEDs are wired the other way around and connected to V_{CC} via a resistor. It is also possible to drive relays with the ULN if the power supply to this IC is connected to 12 V. The only purpose of this schematic is to provide a straightforward example. Very likely you already have a large number of simple AVR schematics in your archive that you can use as well.